Modeling the knowledge contents of CBR systems

Agnar Aamodt

Norwegian University of Science and Technology,
Dept. of Computer and Information Science
N-7491, Trondheim, Norway

Abstract

Recent work within the CBR community has studied more systematic means of interrelating different types of domain knowledge, tasks, and methods for building and maintaining CBR systems. This paper reviews work from the knowledge acquisition community, targeted at methodologies and tools for analysis, modeling, and maintenance of knowledge components. It is argued that the knowledge level is the appropriate level for describing the behavior of an intended CBR system, and for identifying the contents of its knowledge components. A framework is outlined in which a knowledge-level modeling methodology is adapted for the modeling and maintenance of CBR knowledge contents.

1. Introduction

Over the years the CBR community have developed a variety of methods, aimed to address different application tasks, and making use of different types of knowledge. Knowledge needed for the CBR process as such has been identified in terms of four *knowledge containers* (Richter, 1995). These are the vocabulary that describes the domain, the set of cases described in this vocabulary, the similarity assessment knowledge, and the solution transformation knowledge (i.e. adaptation knowledge). Extensions to this set has subsequently been suggested, in particular the addition of a separate container for maintenance knowledge (Patterson, 2001). Many CBR methods rely on a body of general domain knowledge to assist the CBR steps, its primary role being to produce explanations and justifications as part of the reasoning (e.g. Porter, 1990; Leake, 1993; Aamodt, 1994). When this type of knowledge becomes a substantial part of a system's knowledge, modeling and maintaining it call for methods from other areas of AI than those specific to CBR. In addition, current case-based systems are to an increasing degree addressing problems that need richer case contents and more complex case structures (Aha and Wettscherek, 1997; Aha, 2000). This calls for more systematic methods for case base knowledge analysis and modeling, i.e. methods beyond the ad-hoc approaches typical of today.

The knowledge acquisition community has over the last decade had a strong focus on frameworks, methodologies, and tools for characterizing various types of knowledge, their interrelations and their roles in problem solving and learning. A recent trend has been to develop reusable libraries of generic knowledge modeling components, in order to speed up knowledge base development and maintenance, and to facilitate sharing of knowledge bases (Breuker and Van de Velde, 1994; Motta et.al., 1999). Recent work within the CBR community has also looked more deeply into systematic methodologies and support tools for identifying, analyzing, designing, and maintaining the constituents of a CBR system. These research efforts have either taken a systems-structural, implementation-directed or tool-driven perspective (Smyth and Cunningham, 1997; Leake and Wilson, 1998; Leake and Wilson, 1999), or a more generic software architecture or systems development perspective (Plaza and Arcos, 2000; Bergmann et.al., 1997). The work outlined in this paper provides a complementary view by focusing on the analysis and description of knowledge *contents* rather than organizational, implementational or architectural issues. This is done by taking a *knowledge-level* view of the modeling and maintenance process, leaning on results from parts of the knowledge acquisition community, and trying to see how it may can be applicable to CBR systems. Past research within the CBR community on applying knowledge-level analysis to the CBR process (e.g.. Armengol and Plaza, 1993, Althoff and Aamodt, 1996, Fuchs and Mille, 1999) is also related to the work reported here, although it had a different goal than CBR systems modeling and maintenance.

The paper first summarizes the knowledge-level account, and how this has lead to a set of knowledge

modeling methods for knowledge-based systems. On that basis, steps towards a methodology that takes a knowledge-level approach to CBR knowledge container development and maintenance are suggested. The term "knowledge" – as used in this paper – refers to all types of explicitly represented structures on the basis of which a system is able to perform reasoning. In general there are three main types of knowledge we will consider, as elaborated in the next section: Task knowledge, Method knowledge, and Domain knowledge.

2. Knowledge-level modeling

Research within the knowledge acquisition community has produced several methodologies and techniques for describing knowledge at a conceptual, implementation-independent level. Influential examples are the CommonKADS methodology (Breuker and Van de Velde, 1994), the Components of Expertise framework (Steels, 1990), the Generic Tasks approach (Chandrasekaran, 1992), Role Limiting Methods (McDermott, 1988), and the Method-to-Task approach underlying the PROTEGE systems (Musen, 1989). Work in order to unify several of these methodologies has been a focus of several groups, as exemplified by the multiple perspective approach of the KREST methodology (Steels, 1993), and by the generality strived for in CommonKADS (Wielinga et. al., 1992). All these approaches have a common feature, they view knowledge modeling - at least partly - from what is claimed to be a knowledge-level perspective. However, the term knowledge level has become used is slightly different meanings and needs an elaboration.

In Newell's paper (Newell, 1982) the *knowledge level* was proposed as a distinct level of description of computer systems, defined to lie above the level of data structures and programming languages. The latter was referred to as the *symbol level*. In Newell's framework, each computer system level has a medium of expression, identifying what is being processed at that level. Each level further has a behavioral law, which determines how the processing is done, and which enables explanation and prediction of system behavior at that level. The knowledge level has knowledge, in terms of goals and means to obtain them, as its medium, and what is called the "principle of rationality", as its behavioral law. A system is described at the knowledge level as an intelligent agent with its own goals and with knowledge of how to achieve its goals. The principle of rationality states that an agent always will use its knowledge in a way that ensures the achievement of its goals - provided the agent has the knowledge needed. At the symbol level, the medium is symbols (data structures and programs), and the behavioral law is sequential interpretation of program procedures.

The knowledge level enables a system (existing or anticipated) to be described in terms of what it does and why it wants to do it, completely independent of implementational constraints. Hence it is a level applicable to any agent to which it makes sense to ascribe knowledge and rationality (e.g. humans, some animals, some computer systems, ...). The problem with using the knowledge level in this sense, for the modeling and design of computer systems, is that it has no a priori structure. Hence, the knowledge level in this sense cannot be used directly to analyze and structure knowledge. Further, the principle of rationality assumes an ideal rational agent, not bounded by physical or temporal constraints. The knowledge level in its "pure" form is therefore not particularly useful for structuring a knowledge modeling effort. This has lead to modifications of the original knowledge level notion defined by Newell, into a more operational notion of the knowledge level. This may be viewed as moving the knowledge level slightly in the direction of the symbol level. Terms used to characterize this "intermediate level" include the "knowledge use level" (Steels, 1990), and "knowledge level architecture" (Sticklen, 1989). It also includes introducing the notion of "tractable rationality" (Van de Velde, 1993), as a means to deal with the pragmatics of real world situations as opposed to Newell's ideal, unbounded rationality. So, when referring to knowledge-level architectures, methodologies, approaches, etc. within the knowledge acquisition and modeling community, it is this kind of slightly operationalized specialization of Newell's original idea that is meant.

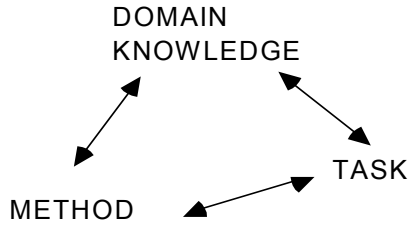

Figure 1: Knowledge perspectives

Given this refined notion of the knowledge level, a consensus seems to have been established that knowledge should be grouped into three main types, or viewed from three perspectives: *Task knowledge*, *Method knowledge*, and *Domain knowledge* (see figure 1). Task knowledge models what to do, usually in a task-subtask hierarchy. Tasks are tightly connected to

goals, and sometimes used interchangeably. A task is defined by the goals that a system tries to achieve. Method knowledge describes how to do it, i.e. a method is a means to accomplish a task (e.g. to solve a problem). Domain knowledge is the knowledge about the world that a method needs to accomplish its task. Examples are facts, heuristics, causal relationships, multi-relational models, and – of course – specific cases. The term "domain knowledge" is not a particularly good term, however, since task- and method knowledge often is domain specific as well. It is hard to find a better term to indicate this type of knowledge, however, although 'object knowledge' 'application knowledge' and just 'models' have been proposed. We will stick to domain knowledge, but bear in mind that the other knowledge types are not necessarily domain independent. Although the in-principle decomposition at the top level into these three knowledge types is agreed upon, the naming of them, and their subdivision and interrelationships are to a large degree what characterizes each specific knowledge-level modeling methodology

By the early nineties, some methodologies and tools had been developed to support this type of knowledge level modeling. They range from relatively general toolboxes to strongly methodology-driven work-benches, usually including libraries of reusable modeling components. Some of these approaches are aimed at knowledge level modeling only (e.g. Breuker and Van de Velde, 1994), while others attempt to provide a bridge to a symbol-level realization – i.e. implementation - as well (e.g. Klinker et al., 1991; Linster, 1992; Steels, 1993). At the basis of several of these methodologies is the view of problem solving as a model construction process.

3. The model construction view of problem solving.

The model construction view states that problem solving is the process of moving from a model instance that describes a problem to be solved, pushing this model through a set of problem solving states, finally ending with a version of the initial model instance that also contains the solution to the problem (Van de Velde, 1993; Clancey 1992). Model instances, sometimes referred to as *case-models*, are state descriptions that contain information about the current state of the world. This also includes the current task that has to be – or is being - accomplished. The role of domain knowledge is to enlarge the case-model, controlled by a suitable method for the task/subtask in question. Characteristic for the view of problem solving as model construction is that the entire case model is considered at each problem solving step. This is different from searching for just a particular value (i.e. a solution). This view therefore fits very well with the CBR problem solving cycle, where the initial case (the problem to be solved) is an instantiation of the generic case model that becomes enlarged through influence from retrieved cases and adaptation knowledge until it contains a solution that satisfies the initial task requirements. This is also advocated by the framework of Plaza and Arcos (2000). Retainment sets up an additional task, which takes the final state of the problem solving case, and constructs a case to be integrated into the case base. Learning can also be viewed as a type of problem solving, i.e. solving a learning problem. This unified view to problem solving and learning also opens up for tightly integrated problem solving and learning architectures (Van de Velde and Aamodt, 1992) where CBR would be one type of method. Adopting the model construction view of problem solving, justified by its easy match with CBR problem solving, opens up for making use of knowledge modeling and maintenance methods that are based on this view.

4. The Components of Expertise framework adapted for CBR

An example of a knowledge level modeling framework, that also incorporates a model-construction view as described above, is Components of Expertise (Steels 1990, Steels 1993). This was one of the methodologies that contributed to the CommonKADS methodology. We have taken initial steps in adapting it for CBR. To get an understanding of the generality of the framework, it is presented in a general way, referring to specific CBR processes as comments and annotations.

The three main types of knowledge (see Figure 1) are in this framework referred to as *tasks*, *methods*, and *models*. So, domain knowledge is referred to as models. There are two types of models: Domain models make up the contents of the knowledge base, and consists of all knowledge that is part of long term memory. This means that general domain knowledge as well as past cases will be contained in domain models. (Domain knowledge in the form of past cases was not an explicit part of the Components of Expertise framework as defined by Steels). *Case models*, on the other hand, are problem solving state descriptions and part of the systems working memory, as described in section 3. Tasks and models are structured into *Task Decompositions* and *Model Dependency Diagrams*, respectively. Methods that impose control over task are described in *Control Diagrams*. See Figure 2.

A task decomposition relates a task to its subtasks in a part-subpart hierarchy. For example, a task may be to diagnose a car, with the subtasks to observe symptoms,

to decide tests, to perform tests, to identify likely faults, etc. The leaf nodes in the hierarchy are tasks that are solved without further decomposition. A task decomposition models the task-subtask relation only, it does not say anything about interdependencies between tasks. This is handled by the control methods.

Model dependency diagrams are used to inter-relate the various domain knowledge types that are needed to construct a new case model on the basis of existing case models. As shown in Figure 2 (middle part), a case model is constructed from information provided by the initial case description (case-model-0), possibly additional information by the user, and knowledge from the domain model (domain-model-1). This could for example be a partial model that infers additional problem descriptors. The resulting case model after the first step (case-model-1) becomes input to a subsequent model construction activity (lower circle), which takes another domain model (domain-model-2),

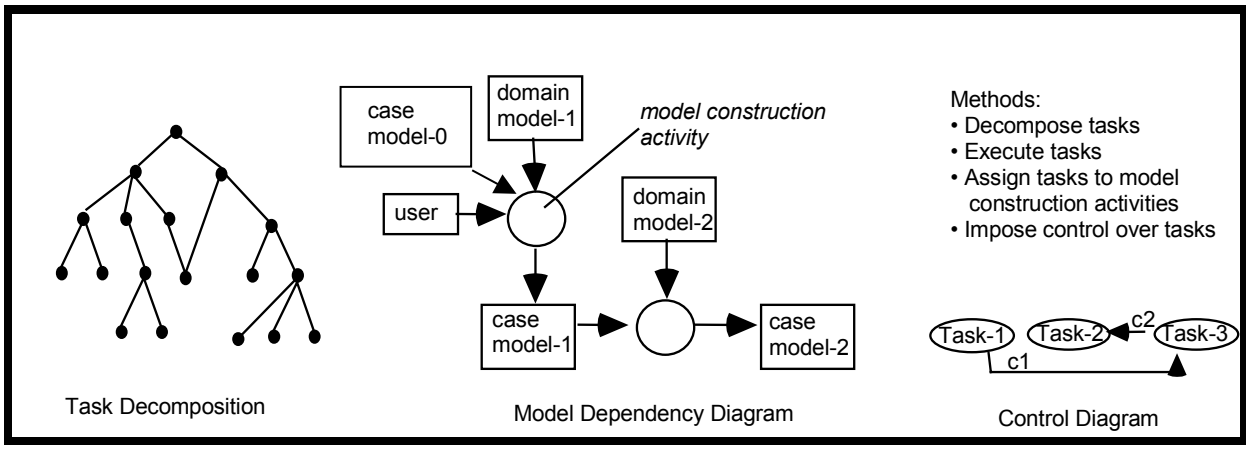

Figure 2: Components of Expertise diagrams

e.g. the case base and supporting general knowledge, and constructs a second case model (case-model-2, e.g. a retrieved case). Model dependency diagrams are organized into abstraction hierarchies. By expanding a model construction activity, model dependencies at a more detailed level are described. Model construction activities will typically map to subtasks in the task decomposition.

Methods are applied to tasks in order to accomplish them. There are four types of methods: A task decomposition method returns a decomposition of the task it is applied to, a task execution method executes a task directly without further decomposition. As mentioned, control methods controls the sequence of subtask execution, based on their dependencies. The final type of method maps tasks to model construction activities (see Figure 2).

The power of using the three perspectives (tasks, methods, and models) for knowledge level modeling lies in the interaction between the perspectives, and the constraints they impose on each other. For example, a task may be decomposed in two principle ways: By a method-oriented decomposition, or by a model-oriented decomposition. In the former, the type of task decomposition method chosen for the task determines subtasks of a task. For example, a common problem solving method called ”cover-and-differentiate” will decompose a task into two sets of subtasks: One which will try to find solutions that cover for the observations made, and another that tries to differentiate between possible solutions in order to find the best one. The method ”hierarchical design” will decompose a design task into a set of course-grained components, which in turn are decomposed into more detailed components, etc. In a model-oriented decomposition, the subtasks of a task are chosen according to what type of domain-models they relate to and the type of case-models they produce. An example would be to decompose a task into a subtask that handles the input of component information, another that deals with process information, etc. The framework opens up for, for example, interrelating cases that cover several subtasks, and to develop problem solving methods and domain models used in retrieval and adaptation suited to each subtask/subcase. A suitable way to integrate knowledge-level and symbol-level modeling will have to be a necessary to be part of the framework (Aamodt, 1995; Winnem, 1996).

5. Beyond the initial steps

Several issues on the current research agenda of the knowledge modeling community are regarded of particular relevance to CBR knowledge modeling, and form a basis for continued research in our group. First, a lot of work is currently being put into the definition and reuse of ontologies, for domain knowledge as well as task and method knowledge (Gennari et. al., 1994; Schreiber et. al., 1995; Fensel et. al., 2000). This will enable the knowledge content of the case vocabulary container, as well as supporting general domain knowledge, to be more easily identified and related to task and method ontologies in a systematic way. In turn this should facilitate the sharing of knowledge for CBR systems development (Bergmann et. al., 1997). Second, the understanding of the role of problem solving methods, and their interrelations with tasks and domain knowledge has increased recently (Aamodt et al., 1992; Benjamins and Pierret-Golbreich, 1996; Motta and Zdrahal, 1998). The explicit modeling of this knowledge type will help in defining the contents of the retrieval and adaptation knowledge containers, and their relation to the domain vocabulary. Finally the case knowledge as such will have to be modeled by taking all the three knowledge types into account, and the knowledge-level analysis will help identifying the relationships between the different part of the case contents.

For CBR, an issue that needs to be looked into in parallel with further development of the framework outlined, is the integration of the knowledge level modeling approach to symbol level design and implementation. Ongoing work in our group is relating the knowledge level framework outlined here to the more symbol-level design framework of Leake and Wilson (1998), and to the development of visualization tools for knowledge-level conceptual modeling as well as symbol-level knowledge representation. Although the framework presented is targeted to the modeling and maintenance of application-related tasks, methods and domain models, the knowledge-level framework also enables the explicit modeling of introspective tasks (Ram and Leake, 1995) and their accompanying reasoning methods and reasoning domain models.

6. Conclusion

Developing and maintaining the contents for CBR containers, and accompanying method and tasks, can benefit from adopting a knowledge-level modeling approach based on state-of-the-art research within the knowledge acquisition and modeling area. The initial framework suggested shows mapping to CBR-specific problems, and points out directions for further research.

References

Aha, D.W., Wettsceherck, D. 1997. Case-based learning: Beyond classification of feature vectors. In *Proceeding of ECML-97, European Conference on Machine Learning.* Prague, 1997. (www.aic.nrl.navy.mil/~aha/ecml97-wkshp/).

Aha, D.W. 2000. Interactive Case-Based Reasoning: Influences, Utility, and Outlook in an Applied World, International Conference on Industrial and Engineering Applications of Artificial Intelligence & Expert Systems. Keynote talk. New Orleans, LA, 21 June 2000. (www.aic.nrl.navy.mil/~aha/).

Aamodt, A., Benus, B., Duursma, C., Tomlinson, C., Schrooten, R., and Van de Velde, W. 1992. Task Features and their Use in CommonKADS. KADSII Report Free University of Brussels, VUB/014. vi+122 pgs.

Aamodt, A. 1994. Explanation – driven-case-based reasoning. In Topics in case-based reasoning, edited by S. Wess et al., Springer Verlag, 274-288.

Aamodt, A 1995. Knowledge Acquisition and Learning from Experience - The Role of Case-Specific Knowledge, In Gheorge Tecuci and Yves Kodratoff (eds): Machine learning and knowledge acquisition; Integrated approaches, (Chapter 8), Academic Press, 197-245.

Klaus-Dieter Althoff, Agnar Aamodt. 1996. Relating case-based problem solving and learning methods to task and domain characteristics; towards an analytic framework. AI Communications - The European Journal of Artificial Intelligence, 9 (3):109-116.

Armengol, E., Plaza, E. 1993. Case-based reasoning at the knowledge-level: An analysis of CHEF. In Proceedings of EWCBR-93, First European Workshop on Case-Based Reasoning. University of Kaiserslautern, 290-95.

Benjamins, R., Pierret-Golbreich, C. 1996. Assumptions of problem solving methods. In Proceedings of the 9th European Knowledge Acquisition Workshop, EKAW-96, N. Shadbolt and K. O'Hara and G. Schreiber (eds), 1-16.

Bergmann, R., Wilke, W., Althoff, K.-D., Breen, S. & Johnston, R. 1997. Ingredients for Developing a Case-Based Reasoning Methodology. In: R. Bergmann & W. Wilke (eds.), Proc. of the Fifth German Workshop on Case-Based Reasoning, 49-58

Breuker, J.A., Van de Velde, W. 1994. CommonKADS Library for expertise modelling. IOS Press, Amsterdam.

Clancey, W. J. 1992. Model construction operators. Artificial Intelligence, 53:1-115.

Fensel, D. Horrocks, I., Van Harmelen, F., Decker, S. Erdmann, M. Klein, M. 2000. OIL in a nutshell. In R. Dieng (ed.), In Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling, and Management (EKAW'00). Lecture Notes in Artificial Intelligence, Springer, 1-16.

Fuchs, B., Mille, A. 1999. A knowledge-level task model of adaptation in case-based reasoning. *In Case Based Reasonning Research and Development, Third International Conference on Case-Based Reasoning*, Seeon Monastery, Germany, July 1999. Lecture Notes in Artificial Intelligence 1650, Springer-Verlag, 118—131.

Gennari, J. H., Tu, S.W., Rothenfluh, T.E., Musen, M.A. 1994. Mapping domains to methods in support of reuse, *International Journal of Human-Computer Studies*, 41, 1994, pp 399-424.

Klinker G., Bohla C., Dallemagne G., Marques D. and McDermott J. 1991. Usable and reusable programming constructs. *Knowledge Acquisition* 2:117-136.

Linster M. 1992. Linking models to make sense to modeling to implement systems in an operational modeling environment. In T. Wetter, K.-D. Althoff, J. Boose, B.R. Gaines, M. Linster, F. Schmalhofer, eds.: *Current Developments in Knowledge Acquisition:* Proc. of the 6th European Acquisition Workshop EKAW'92*,* Springer Verlag, 55-74.

Leake D. 1993. Focusing construction and selection of abductive hypotheses. In *Proceedings of IJCAI 1993,* Chambery, France, Morgan Kaufmann, 24-31.

Leake, D. B., Wilson, D. C. 1998. Categorizing Case-Base Maintenance: Dimensions and Directions. *Advances in Case-Based Reasoning: Proceedings of EWCBR-98*, Springer-Verlag, Berlin. 13 pgs.

Leake, D. B., Wilson, D. C. 1999. Combining CBR with Interactive Knowledge Acquisition, Manipulation and Reuse. *Proceedings of the Third International Conference on Case-Based Reasoning, ICCBR-99*, Springer-Verlag, Berlin. 15 pgs.

Motta, E., Zdrahal, Z. 1998. A library of problem-solving components based on the integration of the search paradigm with task and method ontologies. *International Journal of Human-Computer Studies* 49(4):37-470.

Motta, E., Fensel, D., Gaspari, M., Benjamins, R. 1999. Specifications of Knowledge Components for Reuse. *Eleventh International Conference on Software Engineering and Knowledge Engineering (SEKE '99).*

Musen, M.A. 1989. Conceptual models of interactive knowledge acquisition tools. *Artificial Intelligence* 1(1):73-88.

Newell A. 1982. The knowledge level*, Artificial Intelligence*, 18:87-127.

Plaza, E., Arcos, J.-L. 2000. Towards a Software Architecture for Case-based Reasoning Systems. *Foundations of Intelligent Systems, 12th International Symposium*, *ISMIS 2000*. Ras, Z. W. and Ohsuga, S., (Eds.), Lecture Notes in Computer Science, V1932.

Ram, A., Leake, D. 1995. Learning, goals, and learning goals. In A. Ram & D. Leake*, Goal-driven learning*, MIT press, 1995. Ch.1:1-37.

Richter, M.M. 1995. The knowledge contained in similarity measures. Invited talk at ICCBR-95, Sesimbra, Portugal. (http://wwwagr.informatik.uni-kl.de/~lsa/CBR/Richtericcbr95remarks.html)

Schreiber, A. Th., Wielinga, B. J., Jansweijer, W. H. J. 1995. The KACTUS view on the 'O' word. *In IJCAI Workshop on Basic Ontological Issues in Knowledge Sharing*, 1995. Also in: J. C. Bioch and Y.-H. Tan (eds.). Proceedings 7th Dutch National Conference on Artificial Intelligence NAIC'95, EURIDIS, Erasmus University Rotterdam, The Netherlands, 159–168.

Steels L. 1990. Components of expertise. *AI Magazine*, 11(2), Summer, 29-49.

Steels L. 1993. The componential framework and its role in reusability, In J-M. David, J-P. Krivine, R. Simmons (eds.), *Second generation expert systems*, Springer Verlag, 273-298.

Van de Velde W., Aamodt, A. 1992. Machine Learning Issues in CommonKADS. KADSII Report Free University of Brussels, D 2.11, VUB/002/3.0.

Van de Velde W. 1993. Issues in knowledge level modeling, In J-M. David, J-P. Krivine, R. Simmons (eds.), *Second generation expert systems*, Springer. pp 211-231.

Wielinga, B., Van de Velde, W., Schreiber, G., Akkermans, H. Towards a unification of knowledge modelling approaches. *Proceedings of JKAW-92, Japanese Knowledge Acquisition Workshop*. 1992 (17).

Winnem, O.M. 1996, Integrating knowledge-level and symbol-level modelling: The Creest workbench. Master Thesis, Norwegian University of Science and Technology, Department of Informatics. Trondheim, 114 pgs.